# Image-based Modeling and Rendering

## 5. Sprites and Layered Depth Images

*National Chiao Tung Univ, Taiwan*

*By: I-Chen Lin, Assistant Professor*

# Introduction

- How to “tour” into a still environment without complete geometry info.?
  - Not only a single object.
- We have to handle:
  - Motion parallax
  - Occlusion
  - ……
- We need dense samples for the plenoptic function.
- Otherwise, we can make use of partial depth information.

# Introduction

- Tour into the picture.
- Layered depth images

**Ref:**

• Image-based Modeling and Rendering, SIGGRAPH’99 course notes.

• Y. Horry, K. Aanjyo, K. Arai, "Tour into the picture: Using a spidery mesh interface to make animation from a single image", Proc. SIGGRAPH 97, pp.225–232, 1997.

• J. Shade , S. Gortler , L.-W. He , R. Szeliski, "Layered depth images", Proc. SIGGRAPH'98, p.231-242, July 1998.

# Tour into the picture:

## Using a spidery mesh interface to make animation from a single image

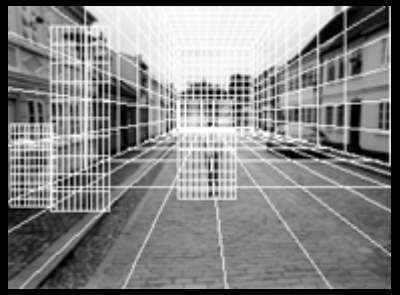
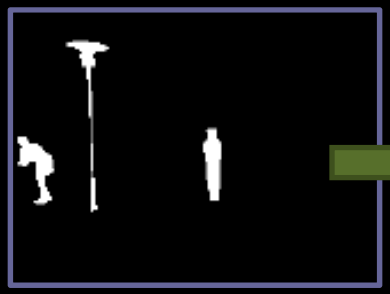
Proposed by

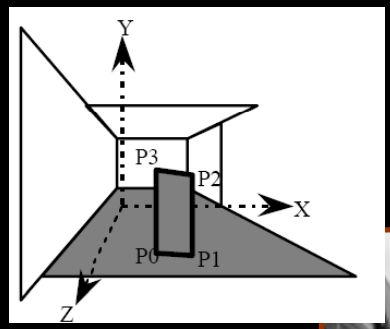
Y. Horry, K. Aanjyo, K. Arai

In Proc. SIGGRAPH'97

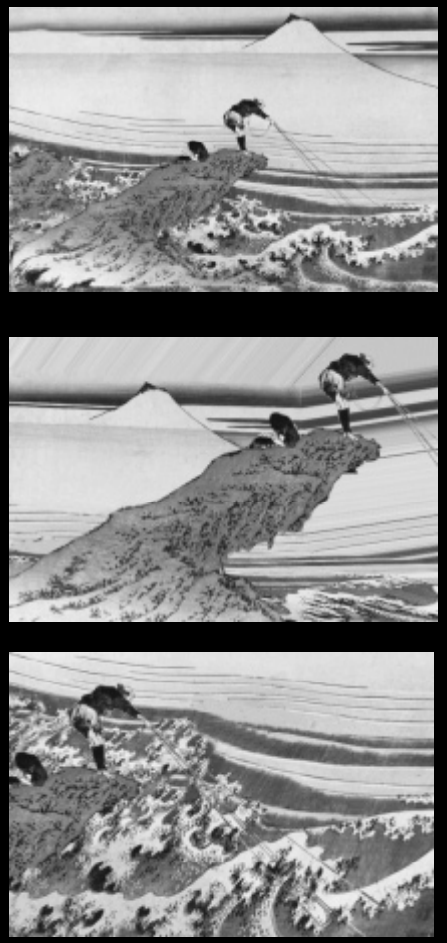
# Tour into the picture





# Tour into the picture (cont.)



# Layered Depth Images

Proposed by

J. Shade , S. Gortler , L.-W. He , R. Szeliski

In *Proc. SIGGRAPH'98*

# Introduction

- How to model a really complex scene? (e.g: trees)
  - Subtle 3D polygonal models
    - High computation complexity
  - Rough polygonal models with texture mapping
    - The problem of resolution
    - Motion parallax (partial)
  - Sprites (extension of texture mapping)
    - Affine or projective transformation
    - Motion parallax (partial)

# Introduction

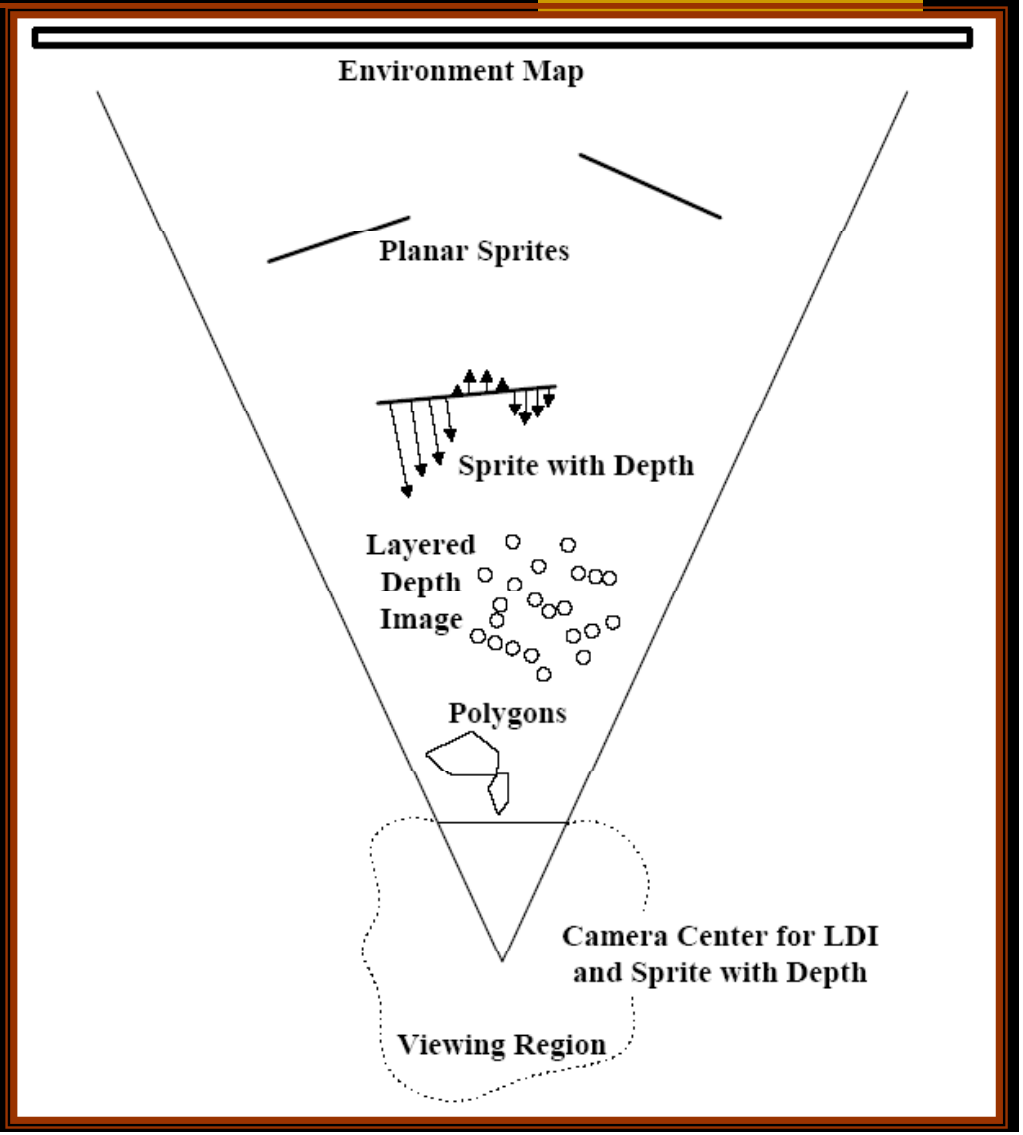
- Why not utilize more depth info.?
- A balance between fidelity (resolution) and computation cost.

- Sprites with depth
  - For smoothly varying surfaces.

- Layered depth images (LDI)
  - For complex geometries.

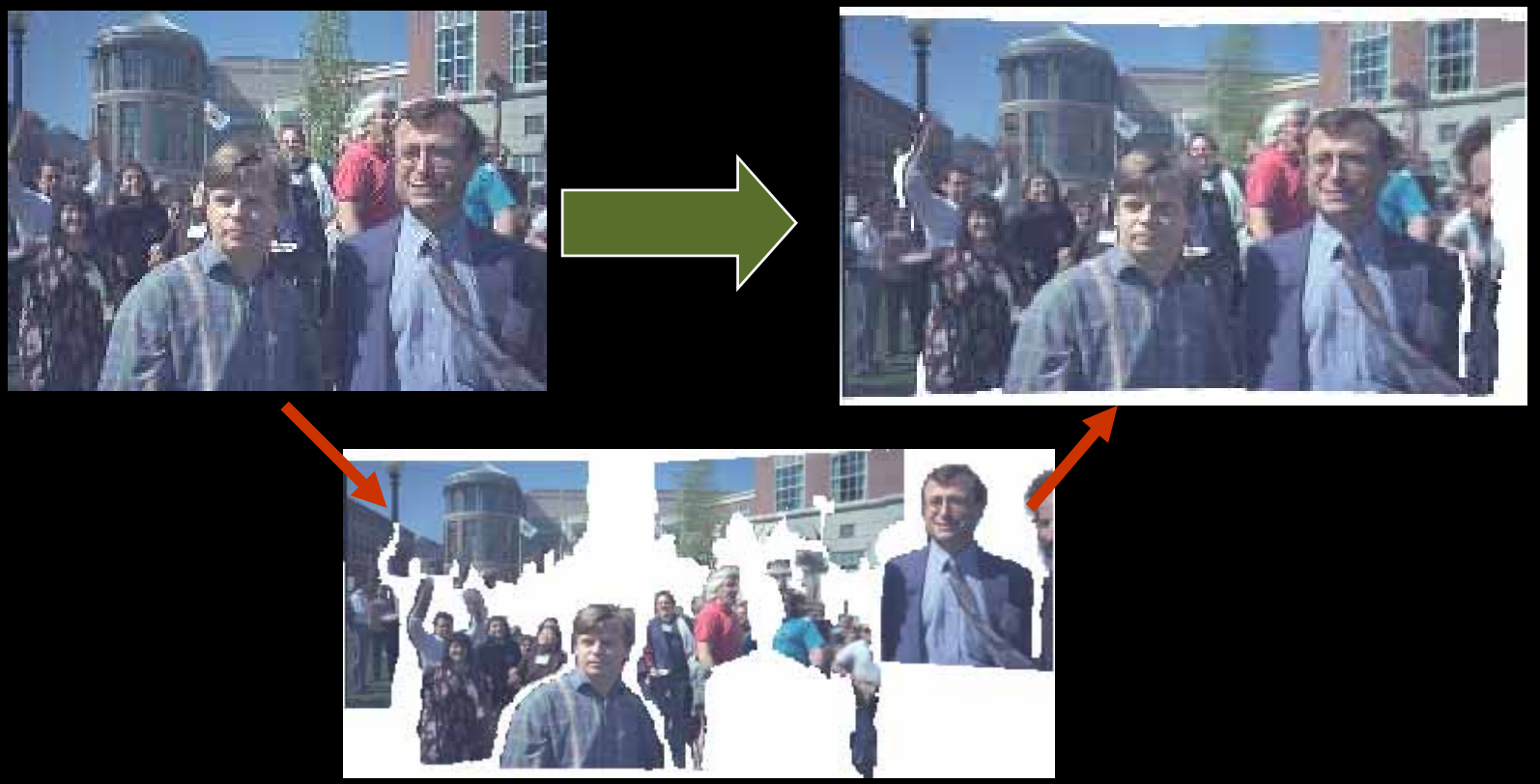
# Introduction

- Choose a suitable solution depending on distance from the camera



# Sprites

- Texture maps or images with transparent pixels.
- Generate new view by warping.



# Sprites


http://www.ozone3d.net/gpu_caps_viewer/

http://sos.planetquake.gamespy.com/archives.php

# Sprites (cont.)

$$\begin{bmatrix} w_1 x_1 \\ w_1 y_1 \\ w_1 z_1 \\ w_1 \end{bmatrix} = C_1 \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

($X$, $Y$, $Z$) world coordinate

($x_1$, $y_1$, $z_1$) the sprite's coordinate

The plane equation of the sprite in the sprite's coordinate:

$ax_1 + by_1 + cz_1 + d = 0$

$$\hat{C}_1 = PC_1, \quad P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ a & b & c & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} w_1 x_1 \\ w_1 y_1 \\ w_1 d_1 \\ w_1 \end{bmatrix} = \hat{C}_1 \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$d_1$ is the scaled perpendicular distance to the plane

# Planar sprites

- The mapping between pixels (x1, y1, d1, 1) in the sprite and (x2, y2, d2, 1) in the output camera's image plane.

$$T_{1,2} = \hat{C}_2 \cdot \hat{C}_1^{-1}$$

$$\begin{bmatrix} w_2 x_2 \\ w_2 y_2 \\ w_2 d_2 \\ w_2 \end{bmatrix} = T_{1,2} \begin{bmatrix} x_1 \\ y_1 \\ d_1 \\ 1 \end{bmatrix}$$

If $d_1$=0 (on the plane)

Drop the 3rd column and row

$$\begin{bmatrix} w_2 x_2 \\ w_2 y_2 \\ w_2 \end{bmatrix} = H_{1,2} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

# Sprites with Depth

- If $d_1$ <> 0

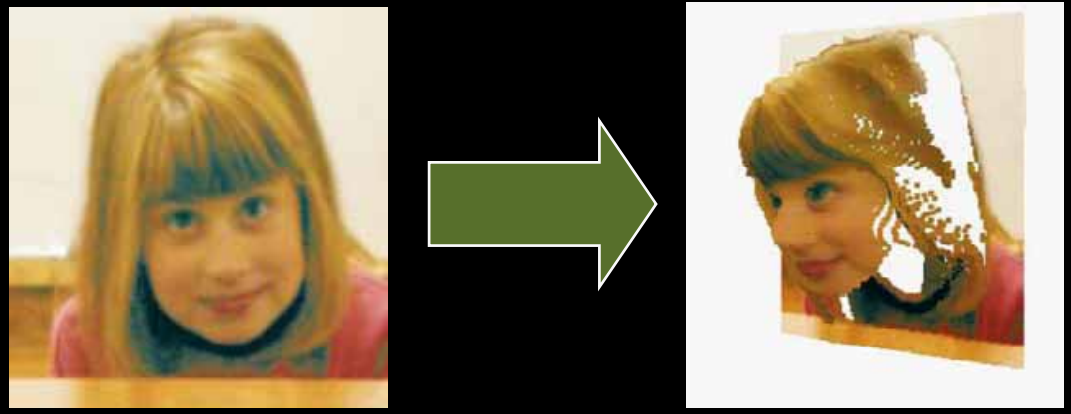
$$\begin{bmatrix} w_2 x_2 \\ w_2 y_2 \\ w_2 \end{bmatrix} = H_{1,2} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} + d_1 t_{c3}$$

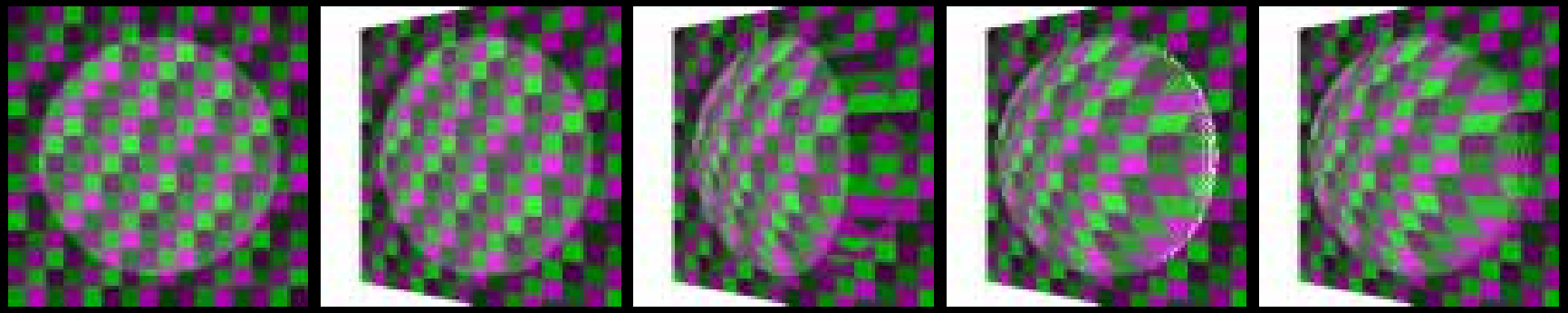
$t_{c3}$ is the 3rd column of $T_{1,2}$

It is also the epipole $e_{1,2}$.

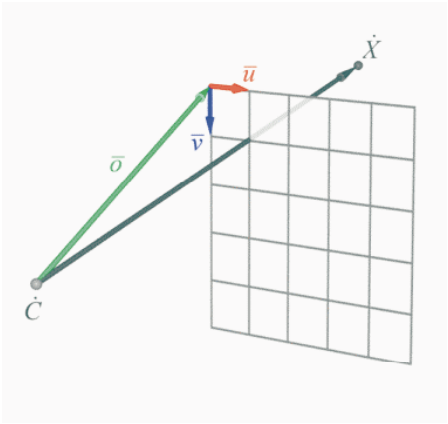
- What's the problem if we apply forward mapping?
  - Holes or gaps.
  - Using splats or interpolation.

# Sprites with Depth (cont.)



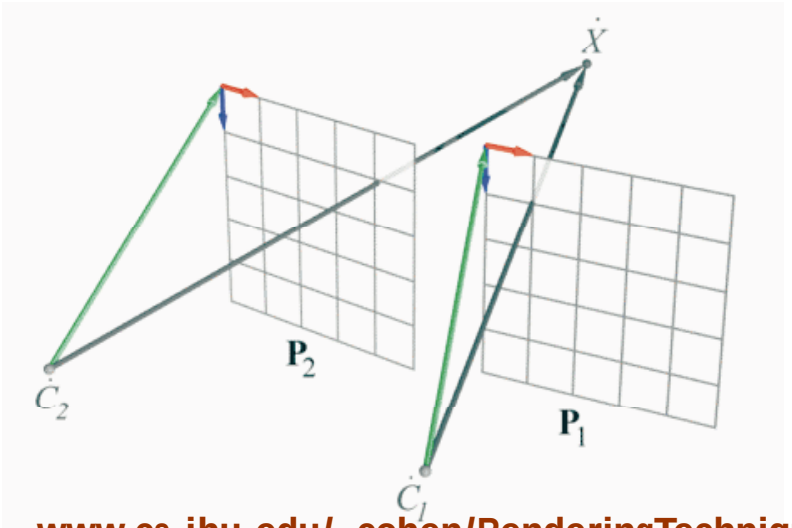


# 3D Warping




$$P = \begin{bmatrix} u_x & v_x & o_x \\ u_y & v_y & o_y \\ u_z & v_z & o_z \end{bmatrix}$$

$$\dot{X} = \dot{C} + t\,P\,\vec{x}$$




$$\dot{C}_1 + t_1 P_1 \vec{x}_1 = \dot{C}_2 + st_2 P_2 \vec{x}_2$$

$$t_2 P_2 \vec{x}_2 = \dot{C}_1 - \dot{C}_2 + t_1 P_1 \vec{x}_1$$

$$\frac{t_2}{t_1} P_2 \vec{x}_2 = \frac{1}{t_1}\left(\dot{C}_1 - \dot{C}_2\right) + P_1 \vec{x}_1$$

$$P_2 \vec{x}_2 \doteq \delta(\vec{x}_1)\left(\dot{C}_1 - \dot{C}_2\right) + P_1 \vec{x}_1$$

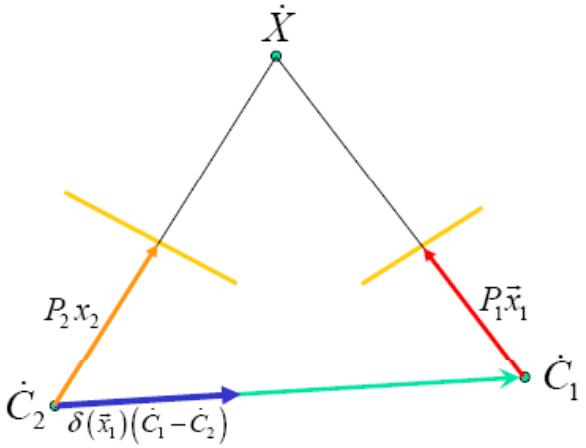
www.cs.jhu.edu/~cohen/RenderingTechniques/Lectures/Image-BasedRendering

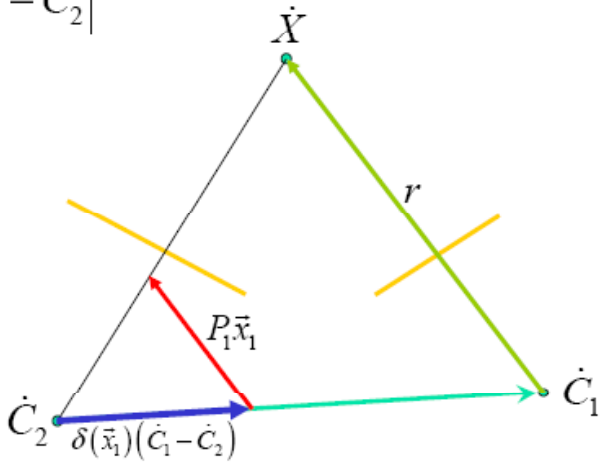
# 3D Warping (cont.)

$$P_2\vec{x}_2 \doteq \delta(\vec{x}_1)\left(\dot{C}_1 - \dot{C}_2\right) + P_1\vec{x}_1$$

$$\frac{r}{\left|\dot{C}_1 - \dot{C}_2\right|} = \frac{\left|P_1\vec{x}_1\right|}{\delta(\vec{x}_1)\left|\dot{C}_1 - \dot{C}_2\right|}$$

$$\boxed{\delta(\vec{x}_1) = \frac{\left|P_1\vec{x}_1\right|}{r}}$$



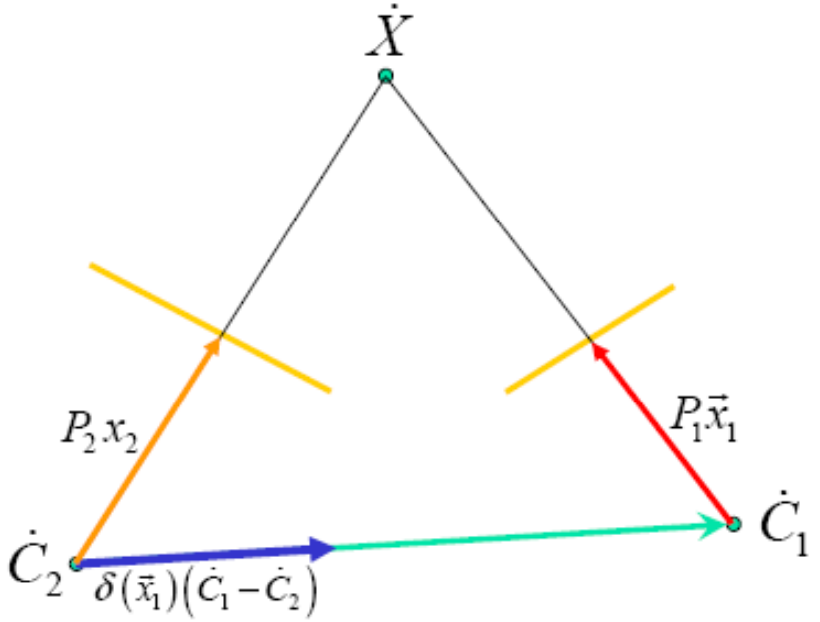
# 3D Warping (cont.)

$$P_2\vec{x}_2 \doteq \delta(\vec{x}_1)\left(\dot{C}_1 - \dot{C}_2\right) + P_1\vec{x}_1$$

$$h\vec{x}_2 = P_2^{-1}\left[\delta(\vec{x}_1)\left(\dot{C}_1 - \dot{C}_2\right) + P_1\vec{x}_1\right]$$

$$h\vec{x}_2 = P_2^{-1}\left[P_1 \quad \left(\dot{C}_1 - \dot{C}_2\right)\right]\begin{bmatrix}\vec{x}_1 \\ \delta(\vec{x}_1)\end{bmatrix}$$

$$h\vec{x}_2 = W\begin{bmatrix}\vec{x}_1 \\ \delta(\vec{x}_1)\end{bmatrix}$$



# Sprites with Depth (cont.)

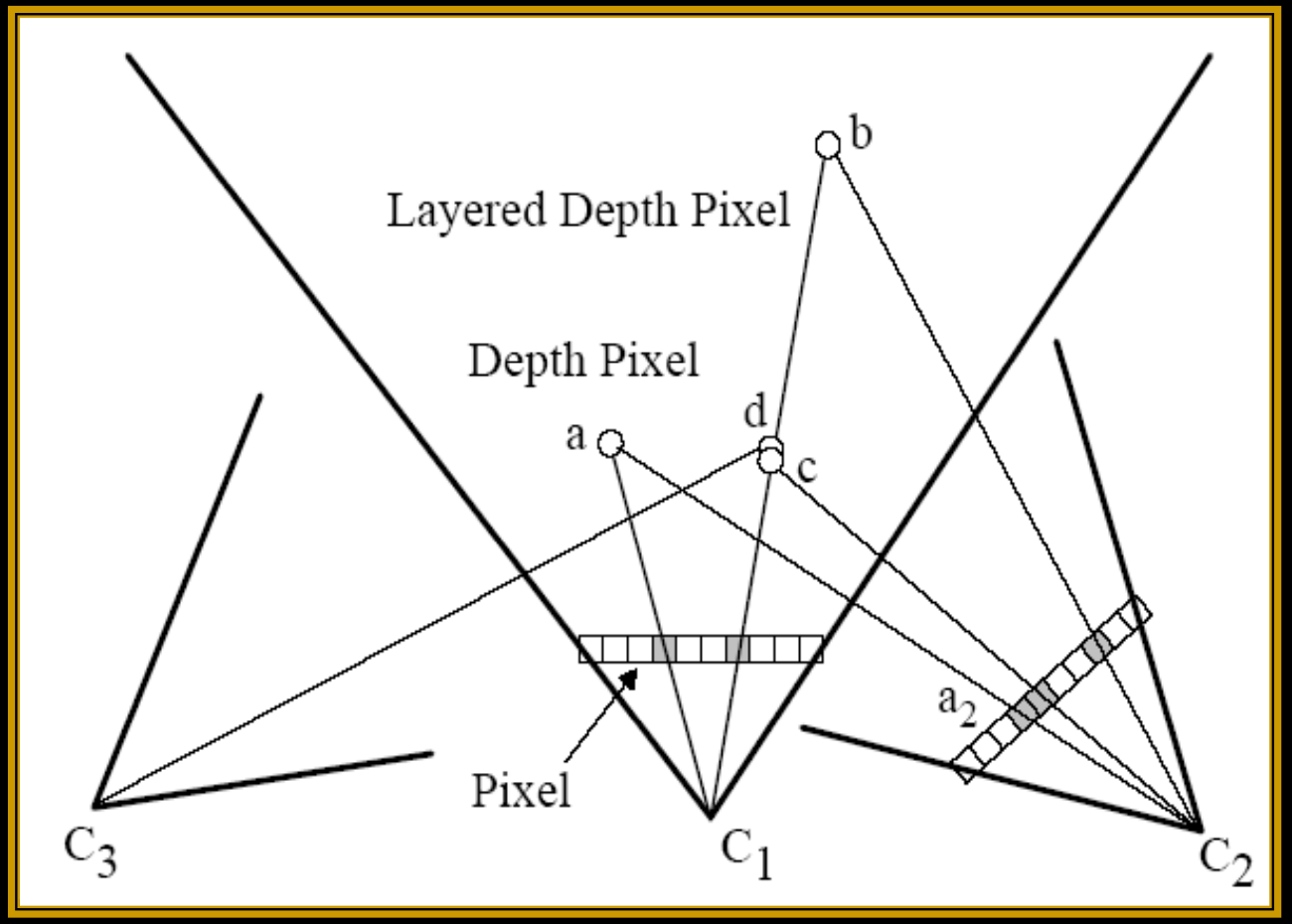
# Sprites with Depth (cont.)

- Depth info.
  - It can be acquired directly in graphics environment.
  - Using stereo reconstruction for real scene.
- What's the limitation of sprites.
  - Local parallax
  - Local and limited disocclusions.
- A more general solution
  - Layered depth images.

# Layered Depth Images

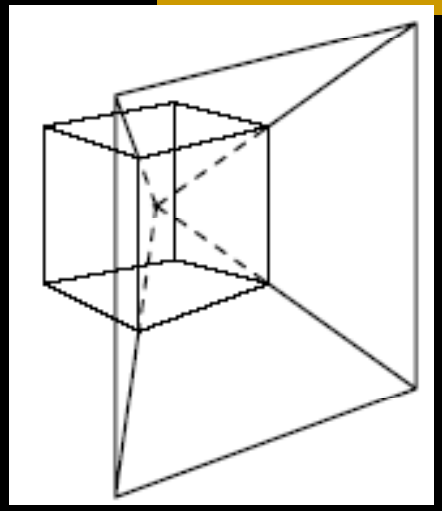
- Pixels also contain depth values with their colors.
- A LDI contains potentially more depth pixels per pixel location.
- **LayeredDepthImage**: [Camera Info. + Array of
  - **LayeredDepthPixel**: [ Array of
    - **DepthPixel** [(R,G,B)+ Z + splat info]
  - ]
- ]

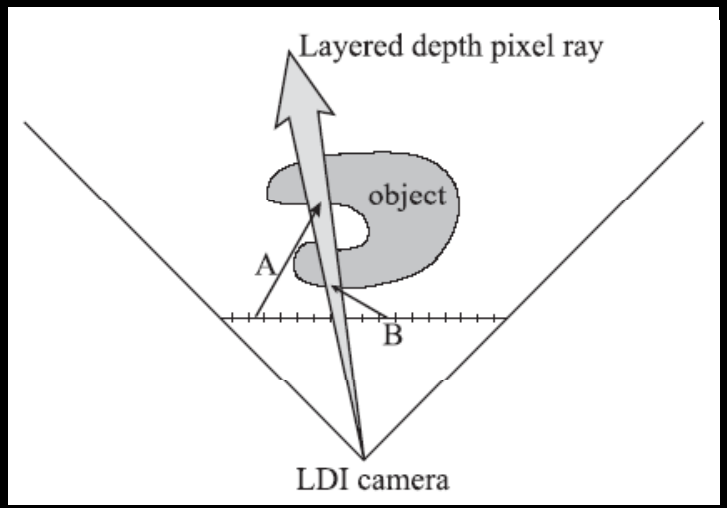
# Layered Depth Images (cont.)



# Construction of LDIs

- LDIs from multiple depth images.
  - E.g. 3D scanned range image
- LDIs from modified ray tracer.
  - More samples ↔ less efficiency.
  - E.g. 6 sides of a cube.
  - Distribution of rays
  - Sampling of depth pixels





# Construction of LDIs (cont.)

- LDIs from real images.
  - Stereo computer vision techniques.
  - Volume data

# Rendering of LDIs

- Suppose that C1 and C2 are the projection matrix of a LDI and the novel view.
  - $(x_1, y_1)$ projected image coordinate
  - $z$ the depth info.

$$T_{1,2} = C_2 \cdot C_1^{-1}$$

$$T_{1,2} \cdot \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_2 \cdot w_2 \\ y_2 \cdot w_2 \\ z_2 \cdot w_2 \\ w_2 \end{bmatrix} = result$$

$$T_{1,2} \cdot \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = T_{1,2} \cdot \begin{bmatrix} x_1 \\ y_1 \\ 0 \\ 1 \end{bmatrix} + z_1 \cdot T_{1,2} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = start + z_1 \cdot depth$$

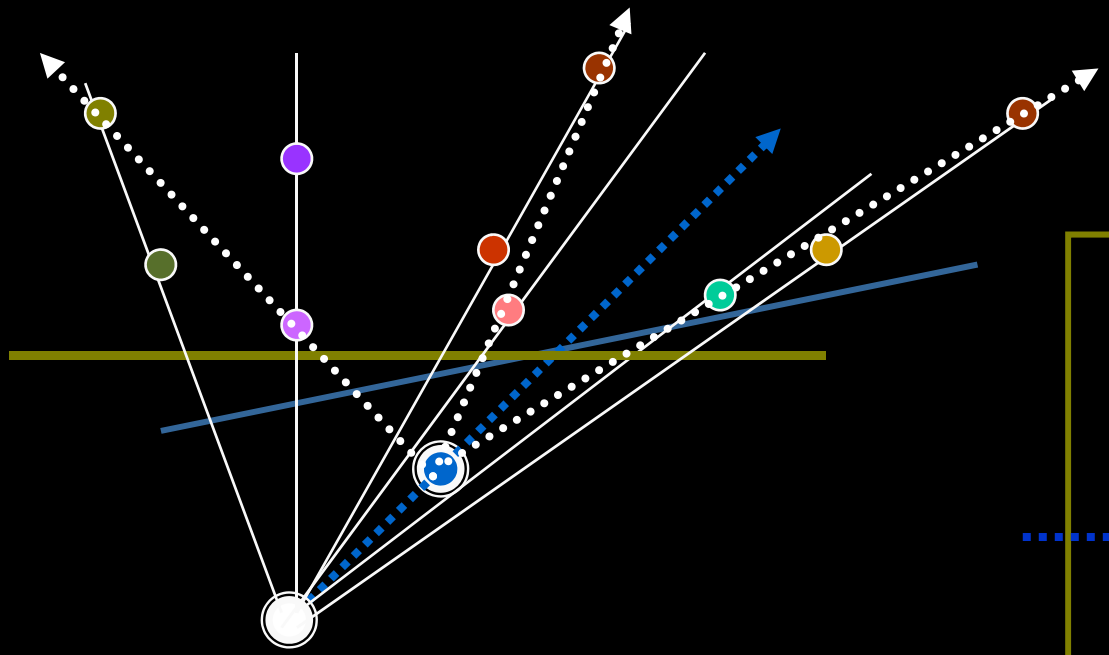
# Rendering of LDIs (cont.)

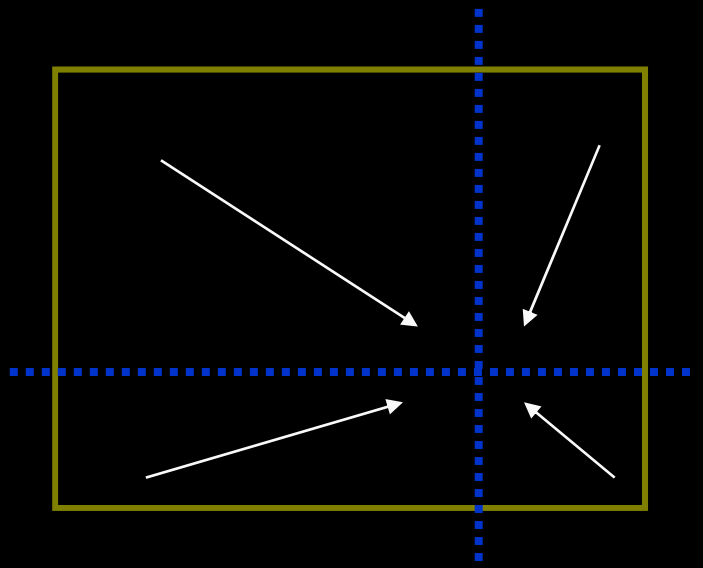
$$T_{1,2} \cdot \begin{bmatrix} x_1 + 1 \\ y_1 \\ 0 \\ 1 \end{bmatrix} = T_{1,2} \cdot \begin{bmatrix} x_1 \\ y_1 \\ 0 \\ 1 \end{bmatrix} + T_{1,2} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = start + xincr$$

- Incremental computation can avoid redundant operations.

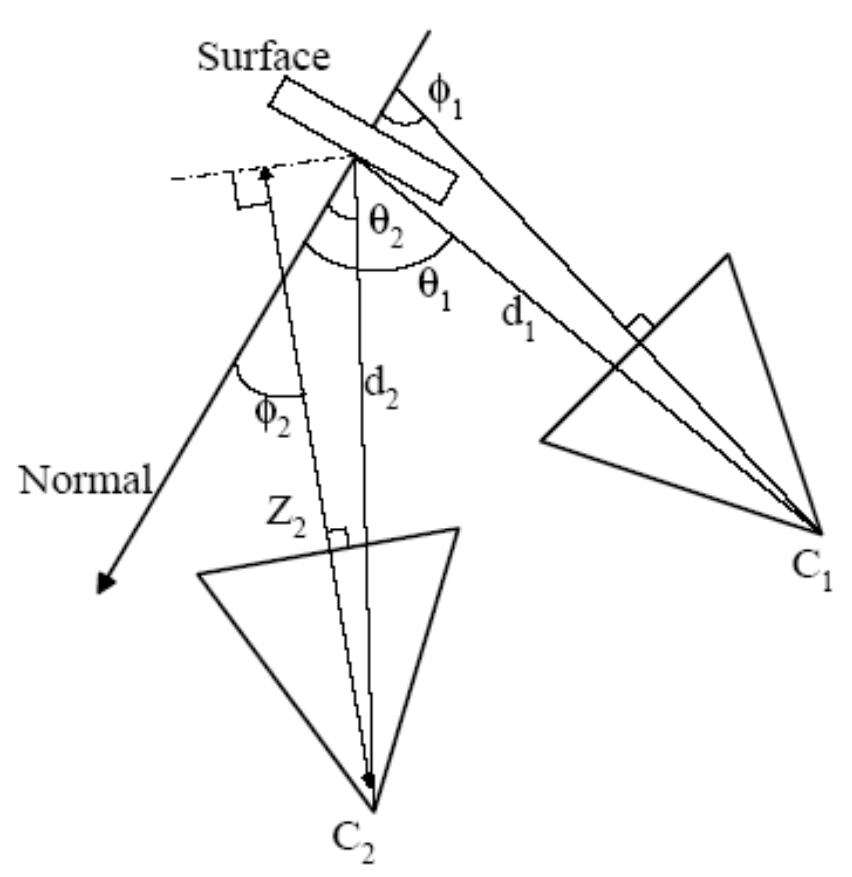
# Hidden surface removal

- Z-buffer……
- Draw without z-buffer (painter's algorithm)





# Splat size

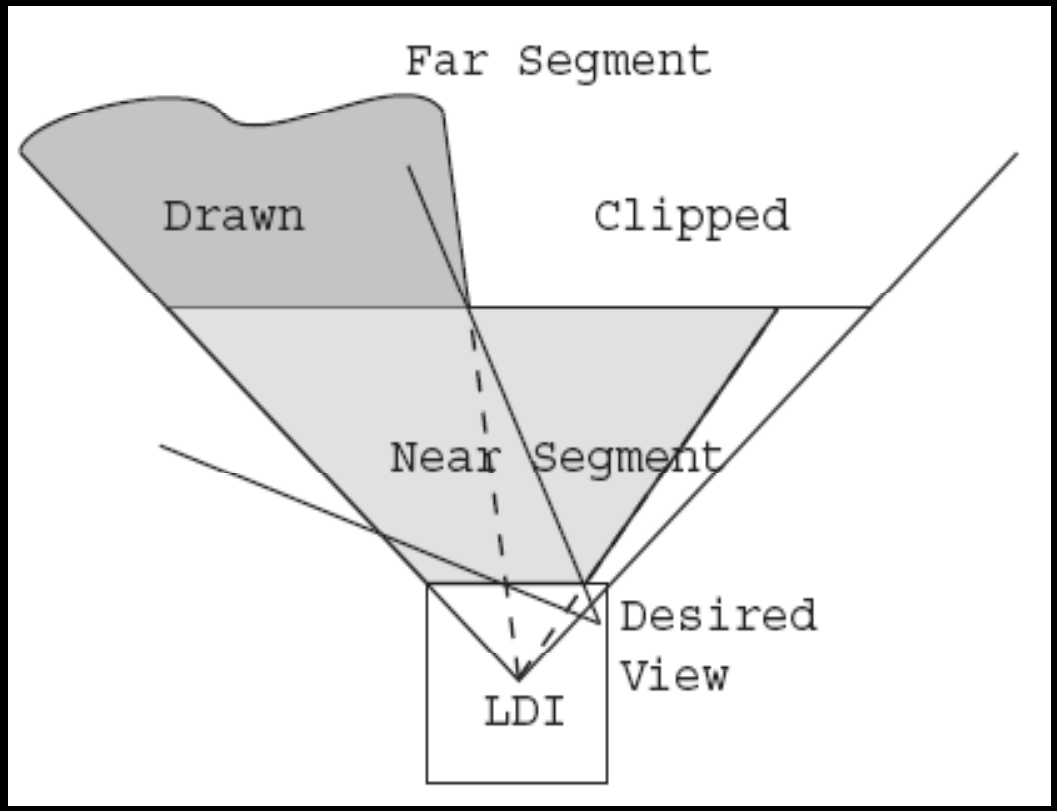



$$size = \frac{(d_1)^2 \cos(\theta_2) res_2 \tan(fov_1/2)}{(d_2)^2 \cos(\theta_1) res_1 \tan(fov_2/2)}$$

$$\sqrt{size} = \frac{1}{d_2} \cdot \frac{d_1 \sqrt{\cos(\theta_2) res_2 \tan(fov_1/2)}}{\sqrt{\cos(\theta_1) res_1 \tan(fov_2/2)}}$$

$$\approx \frac{1}{Z_2} \cdot \frac{d_1 \sqrt{\cos(\phi_2) res_2 \tan(fov_1/2)}}{\sqrt{\cos(\phi_1) res_1 \tan(fov_2/2)}}$$

# Clipping



# Results